Aspect-Oriented Modeling for Representing and Integrating Security Concerns in UML

ABSTRACT

Security is a challenging task in software engineering. Enforcing security policies should be taken care of during the early phases of the software development life cycle to more efficiently integrate security into software. To this end, we present in this paper an aspect-oriented modeling approach for specifying and integrating security solutions into UML design models. The proposed approach covers the specification of aspects, their specialization for a specific design, and their weaving into the base models.

Keywords

Aspect-Oriented Modeling, UML, Security Requirements, Security Enforcement, Access Control.

1. INTRODUCTION

With pervasiveness of computer systems in all aspects of human activities, software complexity is increasing drastically, resulting in the interest on generating robust code from high level design languages like UML. In this context, we propose an approach for representing and integrating security aspects in UML. Our work is part of a research project (MOBS2) on the model-based engineering of secure software and systems. This project aims at providing an end-to-end framework for secure software development that starts from specifying the needed security requirements on UML models and ends with generating secure code. In this paper, we focus on enforcing the needed security requirements using Aspect-Oriented Modeling (AOM) [1].

In fact, AOM has become the center of many recent research activities [3, 4, 6, 7, 11, 15, 16]. AOM allows software developers to conceptualize and express concerns in the form of aspects at the UML design stage, and integrate them into their UML diagrams using UML composition techniques. The usefulness of aspect-oriented techniques for enforcing security requirements in software systems has been already demonstrated in the literature [2, 14]. Though, in despite of the increasing interest, to date, there is no standard language to support AOM, nor a standard mechanism for weaving aspects into the base models.

In this paper, we provide a new approach for specifying security aspects in UML. In addition, our approach allows systematically and automatically weaving security aspects into UML design models and therefore enabling the code generation. By systematically, we mean that our approach supports a complete coverage of the main UML diagrams that are used in software design, and by automatically, we mean that our approach creates the composed new UML models from aspects and original UML models based on formal rules defined in our approach without the intervention of the developer. In the proposed approach, the security expert specifies the needed security solutions as application-independent aspects and how they should be integrated into the design. The developer then specializes the application-independent aspects to his/her design. Finally, our framework injects the application-dependent aspects at the appropriate locations in the design models.

The main contributions of our work are three fold. First, we devise a UML profile that allows security experts to specify security solutions as aspects. In addition, we define a pointcut language to designate the locations where those aspects should be injected in the base models. Second, we introduce a new weaving interface for developers to specialize the generic aspects provided by the security expert. Third, we design and implement a weaving mechanism needed to inject automatically, i.e., without user intervention, aspects into design models. This paper extends the work done in [10] by proposing a more expressive and generic AOM approach for representing aspects, their corresponding adaptations and pointcuts, and their integration into design models.

The remainder of this paper is organized as follows. Section 2 summarizes our approach for specifying and weaving aspects into UML design models. Section 3 introduces an example that will be used to illustrate the usability of the approach. Afterwards, Section 4 presents our UML profile for AOM. Section 5 shows the feasibility of the approach by applying it to the example of Section 3. Section 6 gives an overview of the related work. Finally, we conclude the paper and present our future work in Section 7.

2. MOBS2 FRAMEWORK

Security as a non-functional requirement of the software can be modeled as an aspect. An aspect specifies the adaptations (modifications) that should be performed on the base

SAC'10 March 22-26, 2010, Sierre, Switzerland.

model. A UML profile was developed in our MOBS2 framework such that aspects can be specified by attaching stereotypes, parameterized by tagged values, to UML design elements. The profile is designed to allow as many modification capabilities as possible. These capabilities are presented in Section 4. As part of this UML profile, we developed a high level language to present the pointcuts that specify the locations in the base model where the aspect adaptations should be performed (See Subsection 4.3). The main focus of this paper is to present this UML profile (See Section 4). The process of applying the aspect adaptations to the base model is commonly called "Weaving". In the following, we present a high level overview of the MOBS2 framework for specifying aspects and weaving them into UML 2.0 design models (See Figure 1). This general overview of our approach clarifies the specificities of this UML profile and helps to better understand the usefulness of our approach.

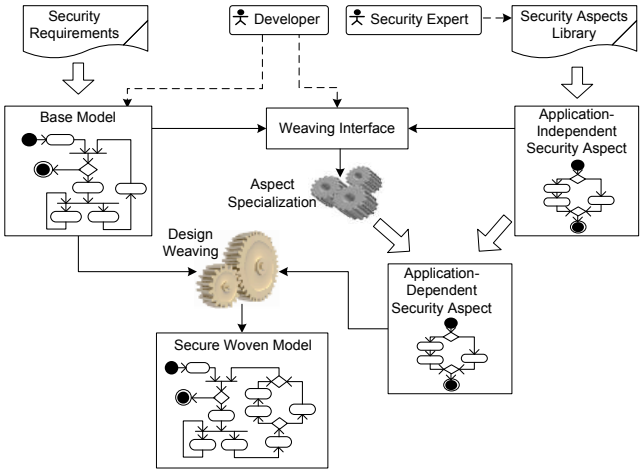

Figure 1: Proposed Approach: Specification and Weaving of UML Security Aspects.

Using this UML profile developed in MOBS2, the security expert has the responsibility of developing the application-independent aspects. By analogy, these aspects are generic templates representing the security features independently from the application specificities and presented in a security aspects library. The developer in turn has the responsibility to specialize these application-independent aspects provided by the security expert according to the application-specific security requirements and needs. Based on the application security requirements and his/her understanding of the application, the developer uses MOBS2 framework to specialize the aspect adaptations and pointcuts to his/her base model elements (See Subsection 5.1).

Based on the pointcuts specified in the aspect by the security expert and specialized by the developer, the MOBS2 framework identifies and selects, without any developer interaction, the join points from the base model where the aspect adaptations should be performed (See Subsection 5.2).

At the end, the MOBS2 framework automatically weaves the above modifications into the base model. To provide a portable solution, we adopted a model-to-model transformation language; the QVT language [9]. For each aspect adaptation, a set of QVT rules are generated by the MOBS2 framework. The order of applying these rules to the base model satisfies the semantics of the original aspect adaptations (See Subsection 5.3).

Before presenting the details of the AOM profile specification, we introduce in the next section an example application that will be used throughout the rest of the paper to illustrate the concepts and the steps of our proposed approach.

3. USE CASE STUDY: RBAC SUPPORT

This section presents a service provider application that will be used to illustrate our approach. The use case consists of enforcing access control to different application resources based on Role-Based Access Control (RBAC) security model. The class diagram of the service provider application is depicted in Figure 2. The class *Client* represents the application's users (e.g., administrator, subscribers, managers). Each type of user has specific privileges. A client accesses the database of subscribers (*ResourceDB*) through an interface *Provision* that is implemented by the classes *SubscriberManager* and *ServiceManager* for manipulating subscribers and services respectively.

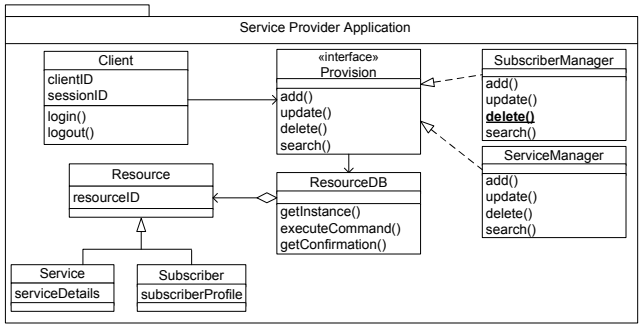

Figure 2: Class Diagram for a Service Provider Application.

Figure 3 represents a sequence diagram specifying the behavior of the method *SubscriberManager.delete()*. The client's permissions must be verified before deleting a subscriber (i.e., only the administrator can delete a subscriber). In this example, we assume that RBAC model should be used to implement the access control.

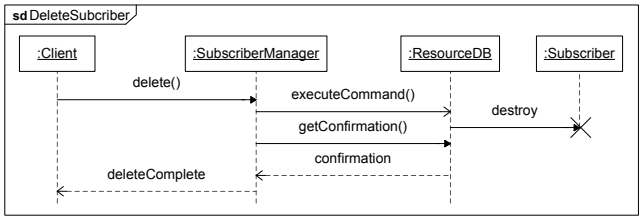

Figure 3: Sequence Diagram Specifying the Behavior of the Method *SubscriberManager.delete()*.

In the following sections, we show how the designer uses our RBAC aspect and MOBS2 framework to weave access control into the base model to check user permissions before deleting a subscriber. We first present the specification of the RBAC aspect using our proposed AOM profile. Then, we present how the RBAC components can be integrated into the application's base model.

4. A UML PROFILE FOR ASPECT-ORIENTED MODELING

An aspect represents a non-functional requirement. It contains a set of adaptations and pointcuts. An adaptation specifies the modification that an aspect performs on the base model, while a pointcut specifies the locations in the base model (join points in AOP) where an adaptation should be performed. In our profile, an aspect is represented as a stereotyped package (Figure 4). For example, Figure 5 shows a partial specification of the RBAC aspect. It is modeled as a package stereotyped ≪*aspect*≫. In the following subsections, we show how adaptations and pointcuts can be specified using our AOM profile.

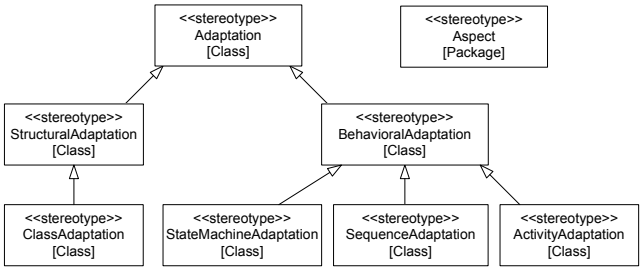

Figure 4: Meta-language for Specifying Aspects and their Corresponding Adaptations.

4.1 Aspect Adaptations

We classify adaptations according to the covered diagrams and the modification rules that specify the effect of adaptations on the base model. UML allows the specification of a software from multiple points of view using different types of diagrams. Unfortunately, most of existing AOM approaches specify aspects within the same modeling view. In this paper, we propose an AOM approach that covers both structural and behavioral views of a system. Note that this does not mean that we cover all existing UML diagrams. Instead, we focus on those diagrams that we believe are the most used by developers: class, sequence, state machine, and activity diagrams. Figure 4 presents our specification of adaptations. We define two types of adaptations: structural and behavioral adaptations.

4.1.1 Structural Adaptations

Structural adaptations specify the modifications that affect structural diagrams. We focus on class diagrams since they are the structural diagrams the most used in the design of a software. A structural adaptation is modeled as an abstract meta-element named *StructuralAdaptation*. It is specialized by *ClassAdaptation* used to specify class diagram adaptations that will contain adaptation rules for class diagram elements (See Subsection 4.2). For example, *RoleAddition* in Figure 5 is a class adaptation (stereotyped ≪*ClassAdaptation*≫) used for the integration of a class *Role* into the class diagram of the service provider application (Figure 2) as well as the adaptation rules that are required to the adoption of an RBAC solution. The definition and the specification of adaptation rules will be presented later in this section.

4.1.2 Behavioral Adaptations

Behavioral adaptations specify the modifications that affect behavioral diagrams. In our approach, we support the behavioral diagrams that are the most used for the specification of a system behavior, mainly, state machine, sequence, and activity diagrams. A behavioral adaptation is modeled as an abstract meta-element named *BehavioralAdaptation*. It is specialized by three meta-elements: *StateMachineAdaptation*, *SequenceAdaptation*, and *ActivityAdaptation* that are used to specify adaptations for state machine, sequence, and activity diagrams respectively. For example, *CheckAccess* in Figure 5 is a sequence adaptation (stereotyped ≪*SequenceAdaptation*≫) defining the adaptation rules required to inject the behavior needed to check user permissions before any call to a sensitive method.

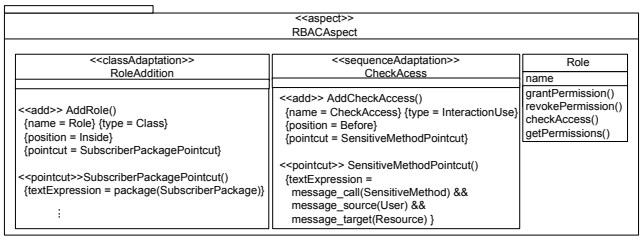

Figure 5: Partial View of the RBAC Aspect.

4.2 Aspect Adaptation Rules

An adaptation rule specifies the effect that an aspect performs on the base model elements. We support two types of adaptation rules: *adding* a new element to the base model and *removing* an existing element from the base model. Figure 6 depicts our specified meta-model for adaptation rules.

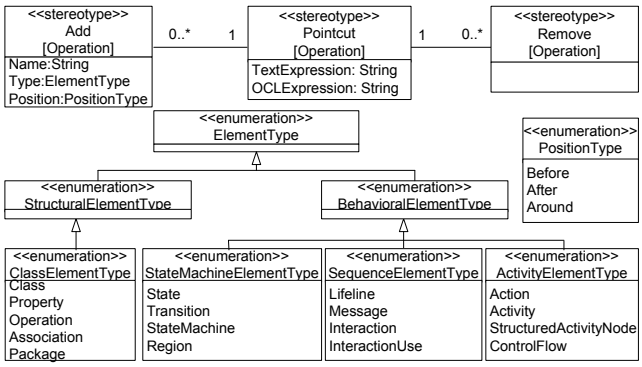

Figure 6: Meta-language for Specifying Adaptation Rules.

4.2.1 Adding a New Element

The addition of a new diagram element to the base model is modeled as a special kind of operation stereotyped ≪*Add*≫. We use the same specification for adding any kind of UML element, either structural or behavioral. Three tagged values are attached to the stereotype ≪*Add*≫:

- *Name:* The name of the element to be added to the base model.

- *Type:* The type of the element to be added to the base model. The values of this tag are provided in the enumerations *ClassElementType*, *StateMachineElementType*, *SequenceElementType*, and *ActivityElementType*.

- *Position:* The position where the new element needs to be added. The values of this tag are given by the enumeration *PositionType*. This tag is needed for some elements (e.g., a message, an action) to state where exactly the new element should be added (e.g., before/after a join point). For some other elements (e.g., a class, an operation), this tag is optional since these kinds of elements are always added inside a join point.

The location where the new element should be added is specified by the meta-element *Pointcut* (Section 4.3). For example, in Figure 5, the operation *AddRole()* stereotyped ≪*Add*≫ is an adaptation rule belonging to the class adaptation *RoleAddition*. It adds a new class *Role* to the package *SubscriberPackage* matched by the pointcut *SubsriberPackagePointcut*. The class *Role* is defined inside the aspect.

4.2.2 Removing an Element

The deletion of an existing element from the base model is modeled as a special kind of operation stereotyped ≪*Remove*≫. The set of elements that should be removed are given by a pointcut expression specified by the meta-element *Pointcut* (See Subsection 4.3). The same specification is used for removing any kind of UML element, either structural or behavioral. No tagged value is required for the specification of a *Remove* adaptation rule; the pointcut specification is enough to select the elements that should be removed.

The proposed profile for the specification of adaptations and their adaptation rules is expressive enough to cover the common AOP adaptations. For example, the profile allows to specify the introduction of a new class to an existing package, a new attribute or an operation to an existing class, or a new association between two existing classes. In addition, we can remove an existing class, an attribute or an operation from an existing class, or an association between two existing classes. As for behavioral modifications, the profile allows to specify the injection of any UML behavior before, after, or around any behavioral UML element matched by the concerned pointcut. Moreover, the proposed adaptation rules are generic; they can be used to specify any security solution for any design. Table 1 summarizes the main adaptation rules that are supported by our approach.

4.3 Pointcuts

A pointcut is an expression that allows the selection of a set of locations in the base model (join points in AOP jargon) where adaptations should be performed. Since the targeted join points are UML elements, pointcuts should be defined based on designators that are specific to the UML language. To this end, we defined in our approach a pointcut language that provides UML-specific pointcut designators needed to select UML join points. Thus, we specify pointcuts as textual expressions using the defined pointcut designators. Due to space limitation, we only present in the following the pointcut designators used to select a class:

- *Class(NamePattern)*: Selects a class based on its name.

- *Inside_Package(PackagePointcut)*: Selects a class that belongs to a specific package matched by *PackagePointcut*.

- *Contains_Attribute(AttributePointcut)*: Selects a class that contains a specific attribute matched by *AttributePointcut*.

- *Contains_Operation(OperationPointcut)*: Selects a class that contains a specific operation matched by *OperationPointcut*.

- *Associated_With(ClassPointcut)*: Selects a class that is associated with a specific class matched by *ClassPointcut*.

Table 1: Supported Adaptation Rules.

UML Diagram	Supported Adaptation Rules
Class Diagram	Adding/Removing a Class Adding/Removing a Property Adding/Removing an Operation Adding/Removing an Association Adding/Removing a Package
State Machine Diagram	Adding/Removing a State Machine Adding/Removing a State Adding/Removing a Transition Adding/Removing a Region
Sequence Diagram	Adding/Removing an Interaction Adding/Removing an Interaction Use Adding/Removing a Lifeline Adding/Removing a Message
Activity Diagram	Adding/Removing an Activity Adding/Removing an Action Adding/Removing a Structured Activity Node Adding/Removing a Control Flow

Those pointcut designators can be composed with logical operators (AND, OR, and NOT) to build other pointcuts.

For example, the pointcut *SensitiveMethodPointcut* in Figure 5 is a conjunction of three pointcuts: (1) *Message_Call-(SensitiveMethod)* selects any message in the base model that calls *SensitiveMethod()*, (2) *Message_Source(User)* selects any message whose source is of type *User*, and (3) *Message_Target(Resource)* selects any message whose target is of type *Resource*. The conjunction of these three pointcuts allows the selection of all message calls to *SensitiveMethod()* from a *User* instance to a *Resource* instance.

The proposed pointcut language is expressive enough to designate the main UML elements that are used in a software design. For example, the pointcut language allows to designate a package, a class, an operation, an attribute, or an association in a class diagram, a message, a lifeline, or an interaction in a sequence diagram, etc. In addition, our proposed pointcut language provides high-level and user-friendly primitives that can be used by the security expert to designate UML elements. In the MOBS2 framework, these pointcuts are automatically translated into OCL expressions that are evaluated on the base model elements to select the targeted join points.

5. PROTOTYPE

To demonstrate the feasibility of our approach, we have designed and implemented a prototype as a plugin to the Rational Software Architect development environment. The plugin uses the AOM profile presented in Section 4 and provides the weaving capabilities needed to weave the aspects

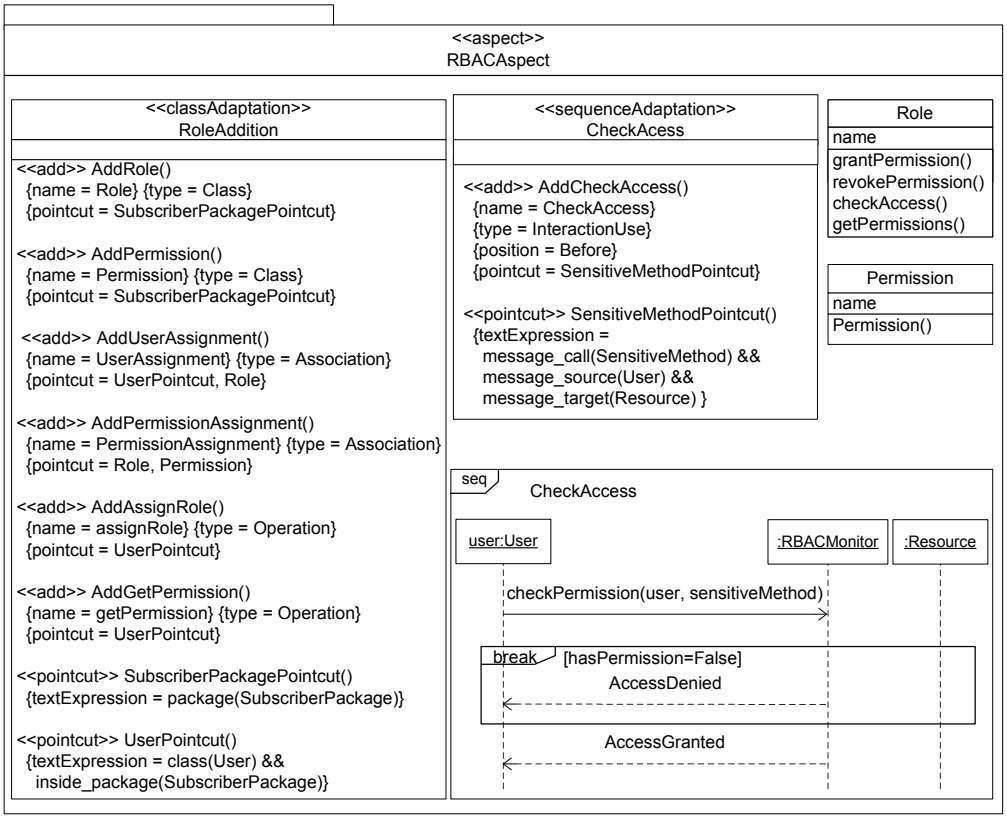

Figure 7: Specification of the RBAC Aspect.

specified by the profile into the base model. The plugin provides also a user interface facilitating the specialization of aspects and their weaving in a systematic way. In this section, we show the feasibility of our approach by illustrating how the prototype can be used by the designer to apply the RBAC aspect (Figure 7) to the base model of the service provider application (Figures 2 and 3). The MOBS2 framework provides the developer with an RBAC aspect designed before hand by the security expert. This RBAC aspect is though application-independent and must be specialized by the developer to the service provider application.

5.1 Aspects Specialization

During this step, the developer specializes the generic aspects by choosing the elements of his/her model that are targeted by the security solutions. The pointcuts specified by security experts are chosen to match specific points of the design where security methods should be added. Since security solutions are provided as a library of aspects, pointcuts are specified as generic patterns that should match all possible join points that can be targeted by the security solutions. To specialize the aspects, we provide a weaving interface in which only the generic pointcuts are exposed to the developers. In our example, the developer maps *SensitiveMethod* to *SubscriberManager.delete()* (Figure 2). The same way, the developer maps *User* to *Client*, *Resource* to *Subscriber*, and *SubscriberPackage* to *ServiceProviderApplication*. Note that *RBACMonitor* in the RBAC aspect is used internally and transparently to the developer to monitor access rights from a client to any class with sensitive methods.

5.2 Join Points Identification

During this step, the actual join points where the aspect adaptations should be performed are selected from the base model. To select the targeted join points, the textual pointcuts presented in Subsection 4.3 need to be translated to a language that can navigate the base model and select the considered join points. In our approach, we use the standard OCL as a query language. We translate textual pointcuts to OCL constraints which serve as predicates to select the considered join points.

5.3 Weaving using QVT Transformations

During this step, the aspect adaptations are woven into the base model at the identified locations according to the specification of the security solution. In the MOBS2 framework, we adopt model-to-model transformation using the standard QVT (Query/View/Transformation) language. We implemented the weaving rules using the Eclipse M2M QVT Operational plug-in [9] on top of the Rational Software Modeler. The input models of the QVT transformation are the base model and the specialized aspect model, and the generated output model is the woven model.

6. RELATED WORK

In the following, we first present the related work on AOM for security before discussing the previous work on weaving aspects into UML models. Regarding the specification of aspects in UML, considerable work has been done in the literature. An overview of existing AOM contributions can

be found in [12, 13]. However, most of existing approaches are programming language dependent and only cover few of AOP capabilities.

Regarding the use of AOM for security, many approaches have been published recently [3, 4, 6, 7, 11, 15, 16]. The following is a brief overview of each approach. [3] proposes an aspect-oriented framework to support the design and analysis of non-functional requirements defined as reusable aspects for distributed real-time systems using UML and formal methods. [4] uses UML diagram templates for modeling access control aspects as patterns. [6] proposes an aspect-oriented design approach for CORBA AC; a reference model for enforcing access control in middleware applications. The RBAC core model is specified as the base model and each RBAC concern is specified as an aspect that crosscuts the base model. [7] proposes an aspect-oriented risk-driven methodology for designing secure applications. After evaluating the application against defined attacks, and if the application presents a security risk, then a security mechanism specified as an aspect is incorporated into the application. [11] extends the UML meta-model with new diagrams to represent access control requirements. [15] proposes an approach for the analysis of the performance effects of security properties specified as aspects. [16] extends the UML-based Web Engineering (UWE) method by specifying the detailed behavior of navigation nodes using state machines. Access control to navigation nodes is specified by refining the default state machines by a state machine modeling the access control rules.

Various approaches have been proposed for weaving UML design models. [17] presents Motorola WEAVR; a tool for weaving aspects into executable UML state machines. However, this weaver is based on the Telelogic TAU G2 implementation, therefore, it is tool-dependent and not portable. [5] proposes a model weaver for aspect-oriented executable UML models. However, the considered join point model intercepts only the interaction between objects and the proposed tool targets only executable UML models. [8] presents XWeave for weaving models and meta-models. XWeave is based on the Eclipse Modeling Framework. Pointcuts used by XWeave are expressed using oAW; an expression language based on OCL.

In conclusion, most of the related work provide specific solutions to specific design choices while our approach targets to provide a generic solution for specifying any design solution.

7. CONCLUSION AND FUTURE WORK

This paper presents an approach for specifying and weaving security aspects into UML design models. For the specification of aspects, we designed a UML profile allowing the specification of common aspect-oriented primitives as well as new primitives. Furthermore, we developed a framework to specialize the aspects and automatically weave the security mechanisms into the base design. By adopting the standard OCL language for evaluating the pointcuts, our approach is generic enough to specify a wide set of pointcut expressions covering various UML diagrams. The adoption of the standard QVT language for implementing the adaptation rules extends portability of the designed weaver to all tools supporting QVT language. In the future, to complete the full life cycle of the software, we will investigate the generation of secure code from woven models.

8. REFERENCES

[1] AOM Website. http://www.aspect-modeling.org/.

[2] R. Bodkin. Enterprise Security Aspects. In *Proceedings of the 4th Workshop on AOSD Technology for Application-Level Security*, 2004.

[3] L. Dai and K. Cooper. Modeling and Analysis of Non-Functional Requirements as Aspects in a UML Based Architecture Design. In *SNPD*, pages 178–183, 2005.

[4] R. France, I. Ray, G. Georg, and S. Ghosh. Aspect-Oriented Approach to Early Design Modelling. *Software, IEE Proceedings*, 151(4):173–185, 2004.

[5] L. Fuentes and P. Sánchez. Designing and Weaving Aspect-Oriented Executable UML Models. *Journal of Object Technology*, 6(7):109–136, 2007.

[6] S. Gao, Y. Deng, H. Yu, X. He, K. Beznosov, and K. Cooper. Applying Aspect-Orientation in Designing Security Systems: A Case Study. In *Proceedings of the International Conference of Software Engineering and Knowledge Engineering*, 2004.

[7] G. Georg, S. H. Houmb, and I. Ray. Aspect-Oriented Risk-Driven Development of Secure Applications. In L. P. E. Damiani, Ernesto, editor, *LNCS*, volume 4127, pages 282–296. Springer-Verlag, 2006.

[8] I. Groher and M. Voelter. XWeave: Models and Aspects in Concert. In *Proceedings of the 10th Workshop on Aspect-Oriented Modeling*, pages 35–40, 2007.

[9] F. Jouault. Eclipse QVT Operational, 2008.

[10] D. Mouheb, C. Talhi, V. Lima, M. Debbabi, L. Wang, and M. Pourzandi. Weaving Security Aspects into UML 2.0 Design Models. In *Proceedings of the 13th Workshop on Aspect-Oriented Modeling*, pages 7–12. ACM, 2009.

[11] J. Pavlich-Mariscal, L. Michel, and S. Demurjian. Enhancing UML to Model Custom Security Aspects. In *Proceedings of the 11th Workshop on Aspect-Oriented Modeling*, 2007.

[12] R. Chitchyan et al. Survey of Analysis and Design Approaches. TR. AOSD-Europe-ULANC-9, 2005.

[13] A. Schauerhuber, W. Schwinger, E. Kapsammer, W. Retschitzegger, M. Wimmer, and G. Kappel. A Survey on Aspect-Oriented Modeling Approaches. Technical Report, Vienna University of Technology, 2007.

[14] J. Viega, J. T. Bloch, and P. Chandra. Applying Aspect-Oriented Programming to Security. *Cutter IT Journal*, 14:31–39, 2001.

[15] M. Woodside, D. C. Petriu, D. B. Petriu, J. Xu, T. Israr, G. Georg, R. France, J. M. Bieman, S. H. Houmb, and J. Jürjens. Performance Analysis of Security Aspects by Weaving Scenarios Extracted from UML Models. *Journal of Systems and Software*, 82(1):56–74, 2009.

[16] G. Zhang, H. Baumeister, N. Koch, and A. Knapp. AO Modeling of Access Control in Web Applications. In *Proceedings of the 6th Workshop on Aspect-Oriented Modeling*, 2005.

[17] J. Zhang, T. Cottenier, A. Berg, and J. Gray. Aspect Composition in the Motorola Aspect-Oriented Modeling Weaver. *Journal of Object Technology. Special Issue on AOM*, 6(7):89–108, 2007.